IBM

Technical Publication

Automation of Logic Page Printing
C. R. Warburton / TR 00.770

January 9, 1961 TR 00.770

AUTOMATION OF LOGIC PAGE PRINTING

by

C. R. Warburton

IBM

IBM
CONFIDENTIAL

Development Laboratory, Data Systems Division
International Business Machines Corporation, Poughkeepsie, New York

TABLE OF CONTENTS

AUTOMATION OF LOGIC PAGE PRINTING

by

C. R. Warburton

INTRODUCTION

The logic diagram is one of the major documents used by the development engineer in producing a new machine. It is also used by the customer engineer to trouble shoot the machine once it is built. The sheet is used to effect additions, deletions, and changes to the hardware used to implement the logic of the machine.

The original production, and later maintenance, of the logic sheet becomes a very tedious and error prone task. In the past the logic sheet was produced by a draftsman working from the designer's sketch. This was a time-consuming process and one which did not lend itself to repeated corrections.

The logic page print program was devised with the idea of producing, on a high speed computer (IBM 705 Data Processing System [1]), systems diagrams which are comparable in layout to those produced by a draftsman, thus fostering the following advantages:

1. Saving on manhours.
2. Speed of preparation (1 min computer time, 1 min of print out [2]).
3. Increased accuracy.
4. Uniform quality of printing, regardless of the number of changes.
5. Ability to get old level pages.
6. Decreased cost of producing page.

---

1. See appendix 3 for definition of machine configurations.
2. See appendix 4 for a more accurate analysis of program running times.

The engineer's logic sketch is drawn on a special logic page form. The information contained thereon is transcribed, key punched, and placed in a central information source called the "logic page master tape". This magnetic tape contains all the pages for a particular machine and is used to supply the information necessary to print out a particular page. This machine-produced page is then used by the designer to make further changes if he so desires. This master tape contains a running account of what each page looked like at each level of development. [3]

The purpose of the logic page print program is to convert a tape record about a particular page, at a particular engineering change level, into the tape record necessary for off-line printing of the logic page.

If the engineer wishes to see what a page looked like at a particular stage of development, he requests this page. A "select" program searches up the master tape for the requested page and picks off the pertinent information at the engineering change level the engineer wishes. This information is placed on a separate tape which is used as input [4] to the print program (figure 1).[5]

The general philosophy behind the program is that it will be just a reproducer of information contained on the master tape; that is, it will produce no new information. The only exception to this rule is the manner in which the lines connecting the logical elements on the printed page shall be drawn. The program tries to route these lines in a neat fashion, making straight line connections where possible, keeping the length of lines short, and keeping the number of bends in the lines to a minimum. The print program is able to successfully route greater than 99.5 percent of the pages which it is asked to lay out. Note: From the statistics readily available it is not possible to determine the exact percentage of program routing failures above the number quoted above. These failures are overcome manually, either by moving blocks around on the page or by decreasing the density of blocks on the page.

GENERAL DESCRIPTION OF PAGE

The logic page can contain up to 45 logical blocks arranged in a 5 by 9 array; up to 5 blocks per row and up to 9 blocks per column (figure 2).[5] The block positions on a page are numbered from 1A through 5I (figure 2) starting at the upper right-hand corner of the page and working down and to the left. Each box contains such information as logical function, transistor type, physical location, engineering change level of the block, and part number of this logical element.

---

3 For further information on the Design Automation System see "The Recording Checking and Printing of Logic Diagrams" - M. Kloomok, P. W. Case, H. H. Graff - TR 00.01110.672.

4 See appendix 5 for the input tape format.

5 See appendix 1.

Each page also contains such information as name of the page, machine type, logic page number, engineering change history, engineers' comments, edge connectors, and Boolean expressions. Physical pin information may be shown on the perimeter of each block. Provision is made to show the name and the logic number for all lines entering or leaving a page.

Space is left between each box so that there is sufficient room to draw the interconnecting lines. A maximum of 7 vertical lines and 10 horizontal lines are allowed between blocks.

PROGRAM DESCRIPTION

We can consider the print program to be broken into 3 parts: (1) setting up of constant information; (2) routing of lines; and (3) miscellaneous. Before going info detail on each of these parts, it should be explained that the image of an entire page is established in a portion of the computer memory area, and it is this image upon which the print program operates when simulating the drawing of a page. One could think of the print program as taking the place of the draftsman, the computer memory area set aside for the page "image" as a blank piece of paper, and the master tape information as the engineer's rough logic sketch.

1. Setting Up of Constant Information

The first thing the program does is to put into the image the information which goes on the first line of the page such as part number, page name, machine number, and logic number. Next, the program draws the outline of a box and puts the appropriate information inside the block. This information consists of circuit symbols, machine features index, voltage levels, physical location, engineering change level, card cap, and configuration; also, a title may be placed over the block. Now the program assigns possible input locations on the left side of the block. The number of possible inputs is specified in the information supplied to the print program. The locations are as follows:

No. of Inputs	No. of Lines Down the Left Side of the Box
1	4
2	2, 7
3	2, 4, 7
4	2, 4, 5, 7
5	1, 2, 4, 7, 8
6	1, 2, 4, 5, 7, 8
7	1, 2, 3, 4, 5, 7, 8
8	1, 2, 3, 4, 5, 6, 7, 8

A record mark is placed in each selected location and identifies possible input points to the program. In other words, if the engineer does not specify that a connection be made to a specific point on the edge of a box, then the print program must make the choice and does so by using the above table. If a possible input point is not used, the record mark prints out as a blank.

The program now places any stub information at the edge of the box. If the block has an extender block associated with it, then two lines are drawn from the bottom of the original block to the top of the extender block. Next, the description of the interconnections between blocks is stored on drums for future use by the routing portion of the program. The process described above is repeated for each block present on the page.

After all of the boxes which appear on the page have been handled, the program operates on the outputs or lines which enter the page on the left. It determines within limits where each of these lines shall appear. Essentially the left edge of the page is broken up into blocks of 18 lines and the outputs within each block are placed opposite, if possible, an input in the same net. The determination of the position of these lines and their associated information (line name and logic number) depends on the position(s), on the page, of the points which a particular line is feeding. For example, if all the inputs which an output is feeding lie above the area in which the output is to be placed, then the program tries to place the output as close to the top of its area as possible.

2. Routing of Lines

The problem of routing the lines between the blocks on the logic page is no simple one and is, in fact, somewhat akin to a maze problem. There are points to be connected together and certain unallowable paths. The position given to a particular line early in the routing process can affect the position of lines routed later, and perhaps cause an impossible routing situation to arise. Also, the neatness of the routed lines is taken into account, but, in times of conflict, is sacrificed to the necessity of finding any solution for a particular configuration of points.

Before explaining the way in which lines are routed, it is necessary to make a few definitions: (Refer to example 1.[6] Note: The example shows only one net.)

a) Net - a group of points which are connected together (points 1 to 10). The print program routes the lines one net at a time.

b) Output point - the origin point of a net (point 1). The output point always lies on the right side of the block (in a particular net) which is located furthest to the right and highest on the page. It is the point which is used to start transcription of a net.

6. See appendix 1 for examples.

c) Input point (s) - the end point (s) of a net (points 2 to 10).

d) Primary vertical line (PVL) - a vertical line lying in a column to the right of the output point and to which the output point can be connected by a single horizontal line.

e) Regular point - an input which lies in a row in which there are no blocks obstructing a line drawn in that row between the column in which the primary vertical line is placed and the column of that input point. The output point is defined as a regular point. (Points 1, 2, 7, and 10 are regulars.)

f) Feedahead (FA) - an input point which lies on a block located in a column to the right of the output point and which is not a regular point (points 3, 4, 5, and 6).

g) Feedback (FB) - an input point which lies on a block located to the left of the output point and which is not a regular point (points 8 and 9).

h) Secondary vertical line (SVL) - a vertical line to which all feedahead points or feedback points in a given column can be connected by means of single horizontal lines.

i) Feedahead or feedback main horizontal line (MHL) - a horizontal line which connects the primary vertical line to the secondary vertical line (s) to its right (or left).

j) Internal coordinate system - used only by the program. Considers the vertical dimension of the page to be the Y axis and the horizontal dimension to be the X coordinate. The origin is located at the upper left corner of the page.

k) Column - for purposes of the routing portion of the program a column is defined to have the same number as the block column immediately to its left.

Note: The terminology defined here is used only in regard to the program.

General Explanation of Routing (refer to example 2)

A very general explanation of the way lines are drawn for a given net is as follows: All regular points are connected to the primary vertical line. Next, all feedaheads in various columns are connected to their respective secondary vertical lines. This is repeated for any feedbacks in this net. The final step is to connect all the secondary vertical lines to the right of the output point to the primary vertical line with a single horizontal line and to connect all the

secondary vertical lines to the left of the output point to the primary vertical line with a single horizontal line.

Essentially the program puts each of the points in a net into one of three categories: (1) regular, (2) feedahead, or (3) feedback as defined previously. It then routes each of these groups of points separately. First, each of the regular points is connected to the primary vertical line by means of a single horizontal line. A horizontal line (called the feedahead horizontal line) is then drawn to connect the primary vertical line to all of the feedahead points. A vertical line called a secondary vertical line is drawn in each column which contains a feedahead point so as to intersect the feedahead horizontal line. A single horizontal line is then used to connect each feedahead point to the secondary vertical line located in its column. The identical process is repeated for all feedback points in a net, except that the feedback horizontal line, which is used to connect the primary vertical line to the secondary vertical line, is drawn to the left of the primary vertical line. One of the main advantages of routing in the above fashion is that a group of common points in a column are fed by the same secondary vertical line.

Using the above explanation as a guide, let us follow through example 2. Points 1, 2, and 3 are called regulars; points 4, 5, 6, and 7 are called feedaheads, and points 8 and 9 are called feedbacks. A primary vertical line is defined whose length is the vertical distance between points (1) and (3) and whose X coordinate is determined by a set of complex criteria which will be more fully explained later (example 2b). Each of the regular points is then connected to the primary vertical line. A horizontal line is then found which will connect all the feedahead points to the primary vertical (example 2b). Single vertical lines (SVL) are then drawn in columns 2 and 3, and a single horizontal line is used to connect each input point to its respective secondary vertical line (2c).

A horizontal line is then found which will connect all the feedback points to the primary vertical line (2d). A single vertical line (SVL) is then drawn in column 5 and a single horizontal line is used to connect each input point (8, 9) to this SVL. The choice of the horizontal line (s) used to connect the secondary vertical lines is made such that the extensions to the secondary vertical lines will be as short as possible. Example 2c redrawn as example 2e does not satisfy this criterion. Also, the primary vertical line will not be extended unless absolutely necessary.

A detailed description of the exact criteria used in the routing portion of the program and further examples of this may be found in appendix 1. A flow chart of the routing is located in appendix 2.

Additional Functions of Routing

Since the program operates on one net at a time, it encounters certain information (or lack of information) which it must save (or generate) at that particular time. Edge connectors fall into the category of information to be stored, (example 3a). Because it is not practical to place them on the branch of a particular line, it was decided to let the program assign a two digit reference number (for example, *1 in example 3b) to each edge connector and to place this reference number next to the output or input of the box which is connected to the branch in question. The edge connector and its reference number are placed at the bottom of the page (example 3b). After all routing of lines is completed, an attempt to move some of the information to the correct block location is made. This will be explained in more detail under section 3 entitled "Miscellaneous".

The program contains the facility to assign those points which were not given a fixed input position at the edge of a block. These points are assigned so as to make a direct connection between the output and the input, if possible. In the case of feedaheads or feedbacks the inputs are assigned so as to keep the length of the SVL's to a minimum.

As in the case of outputs on the left edge of the page, the program must position the inputs for lines going off the page to the right or left. In this case the program tries to assign a Y coordinate such that a straight line connection can be made between the output and this off-page point. If a straight line connection is not possible, then placement is made using shortest distance as the criteria.

Since the scheme used for routing the line does not guarantee a solution (if one exists), the program, if stuck, tries a variation of the original procedure. To bring about this variation it reclassifies all the regular points in a net as feedaheads (or feedbacks). We see in example 4a that point 1 is classified as a regular, and the net cannot be routed. But in example 4b, point 1 is changed to a feedahead and the net is successfully completed. If this alteration of parameters fails to bring about a solution, the program stores the fact that it could not route this particular net and after processing all the nets on a page, starts the page over completely. This time the program tries the nets in a slightly different order; that is, it first tries those nets which it could not route on the first pass. Again, it makes two attempts to route the net. If it does not succeed on the second pass, it makes no further attempt to find a solution. Thus, the program may try to route a particular net up to four times and may attempt to draw the entire page a maximum of two times.

3. Miscellaneous

Once the routing portion is over, the program makes a clean-up pass over the page. This part of the program is concerned with the following problems: (a) movement of physical pins back up next to the block; (b) arrangement

and printing of edge connectors and/or Boolean equations; and (c) printing out of the page.

a) Movement of physical pins

The program searches along the edge of each box on the page looking for an edge connector reference number. When it finds such a number, it searches up the appropriate edge connector (which was stored earlier in the program) and tries to place the reference number and physical pin on the line (example 3C).

b) Arrangement of data

Next the program counts the number of lines of edge connector and/or Boolean equation information which must be printed on the page. It then looks to see if there is at least an equal number of consecutive blank lines available on the page (below the last piece of logical information). If there is enough room, it will print the required information on the original page. If there is not enough room on this page, the program will print all the edge connector and equations on a second "addenda" page.

c) Printing out of the page

Finally the program generates the appropriate control characters so that no time will be wasted in printing the page on the 717 printer. The "image" of the page which has been formed in memory is then written onto magnetic tape as 185 records consisting of 121 characters each, which in turn can be printed out as an "off line" operation. The 717 printer used has a special type set so that such characters as corners (┌ , ┐ , └ , ┘) and horizontal and vertical lines will print out on the computer produced logic page.

Additions to the Program Which Are Now Under Consideration

Reduction of line crossover - in many cases an alternate path could be found which is easier for the eye to follow because it intersects fewer lines.

Spreading out of adjacent lines - many times it is possible to leave a blank line between two adjacent parallel lines. This would make it easier to follow the logical flow.

Placement of lines entering the left edge of the page - the program would decide where on the edge of the page each of the these lines should originate so that the routing criteria could be better met.

Placement of blocks on the page - such a procedure might aid the logical flow and the routing criteria.

Printing of variable length boxes.

Ability to print engineering notes or comments on the page.

CONCLUSION

Let us recall that the general philosophy behind the print program is that it will in general just reproduce the information supplied to it by the engineer. This information consists of physical and logical information about the design of the machine which may be useful both to the designer while he is perfecting his design and to the customer engineer while he is maintaining the machine in the field. The logic page print program has performed under production conditions for the last 1 1/2 years in such a manner as to substantiate the advantages listed in the introduction to this paper.

ACKNOWLEDGEMENTS

Messrs. R. W. Carpenter and S. Sobel contributed heavily to the initial planning and implementation of the program.

APPENDIX 1

Detailed Description of Routing Process

1. Determine the length of the PVL which is the absolute difference between the Y coordinates of the highest and lowest regular points in a net (example 5).

2. Determine in which column the PVL shall be placed.

   a) The PVL will be placed in the column at the right edge of the page if possible (i.e. if it can be moved there without intersecting another block, example 6).

   b) The PVL will be placed as far to the right as possible if:

      (1) there are no input points (regulars or FA) to the left of the output point (example 7). The dotted lines show the routing if an input to block 3B was also included.

      or (2) there are two or more regular input points to the right of the output point.

      and either

      (3) the output was at the left edge of the page.

      or (4) the net is being done over.

   c) If none of the above conditions are met (i.e. the PVL is placed in the column immediately to the right of the output point) then the program tries to route the PVL through any empty boxes in the next column to the right (example 8). In this case, making use of the additional vertical routing space is more important than the slightly

increased length of the horizontal line.

3. Determine the order in which a search will be conducted to find an open path for the PVL within a given column. The order will be:

   a) left to right if the line is entering the page on the left and it is located opposite a block location (example 9). This procedure prevents the cutting off of other inputs to block 5A.

   b) right to left if:

      (1) the line is entering the page on the left and it is not located opposite a block location (example 10). Prevents the blocking off of the output 22. 22. 22. 2.

      (2) the line is going off the page on the right (example 11). Prevents blocking of other outputs from 1A such as B.

   c) alternately as left then right and as close to the center of a column as possible for all cases not included by a or b (example 12). This minimizes the danger of cutting off other outputs and/or inputs.

4. If the PVL is moved to the right into a new column, any FA points which could now be called regulars due to this movement are so included (examples 13 a, b). It will be noted that the input to 5A is a feedahead point in example 13a and is a regular point in example 13b.

5. The program tries 6 or 7 vertical paths depending on whether the line in question is to the left of block column 5 or to the right of block column 1.

6. In the case where the PVL has been moved to the right of the output column and no paths are available in this new column, then the program moves the PVL back to the next column to the left.

7. If no paths are available at all for the PVL or if not all the regular input points could be connected to the PVL (example 4a), then the net is rerouted with all regular points changed to FA points (example 4b). If this technique does not work then this net is stored along with any others which couldn't be done on this page, and the program tries the "can't route" nets first.

8. Determine how far the PVL can be extended toward the top and bottom of the page. This information would only be put to use in a case such as example 15 where the primary vertical line turns out to be a point and the line must be routed around block 2C. Note the horizontal line which bends just above the top of column 3.

Selection of a Feedahead or Feedback horizontal line.

1. Determine the "highest low" point and the "lowest high" point of all the regular and feedahead, or regular and feedback points (example 16). The lowest point in column 3 is point 2, in column 4 is point 4, and in column 5 is point 6. The highest of these low points is point 4. In a similar manner the lowest high point is point 1.

2. Determine a median point between the "highest low" and the "lowest high" points. Thus the median point of example 16 is $\frac{|Y4 - Y1|}{2}$, which lies between 1 and 4. This is used as a starting point to search for the feedahead and/or feedback horizonal line which will connect the PVL to all of the SVL which are either to the right or to the left of the output point.

3. Starting at the median point the program searches:

   a) up the page until it finds a free path or until it reaches the highest point to which the PVL can be extended.

   b) down the page until it finds a free path or until it reaches the lowest point to which the PVL can be extended.

4. The program chooses for the horizontal line, the line closest to the median point. If both of the horizontal lines found in step 3 are equidistant from the median point the program chooses:

   a) the higher line if it lies within the limits of the PVL.

   b) If the higher line lies off the end of the PVL then it chooses the lower line if the lower line lies within the limits of the PVL.

   c) If neither conditions a or b are met, then it chooses the horizontal line closest to an end of the PVL.

   In other words it chooses a horizontal line such that the PVL need be extended as little as possible. In example 14, the program chose the solid rather than the dotted path, since it chose as a feedahead horizontal line the lower path which extended the PVL the least.

5. The length of the horizontal line is defined as the absolute difference between the X coordinate of the PVL and X coordinate of the right edge of the furthest FA column.

Selection of Secondary vertical lines

1. All FA columns are handled separately from all FB columns in a net.

2. Determine the length of SVL which is:

   a) the difference between the highest and lowest FA points in a particular column (example 17a, points 3 and 4)

   b) the difference between the Y coordinate of the feedahead or feedback horizontal line and the Y coordinate of either the highest or lowest FA point in this column (example 17b, points 3 and 4) if the main horizontal line does not intersect the SVL.

3. Determine the order in which to search for possible paths for the SVL.

   a) The program tries 6 paths from right to left if the SVL lies in the column at the right edge of the page.

   b) Anywhere else on the page the program starts in the middle of a column and alternates the paths tried.

4. Determine if the possible path selected for the SVL is open. The conditions triggered by an affirmative or negative answer to this question are more easily seen by referring to the flow chart (appendix 2 - "Determination of length and path of SVL").

5. After all FA and then all FB columns in this net have been completed, all lines and special characters (representing corners and branches) are placed in the image.

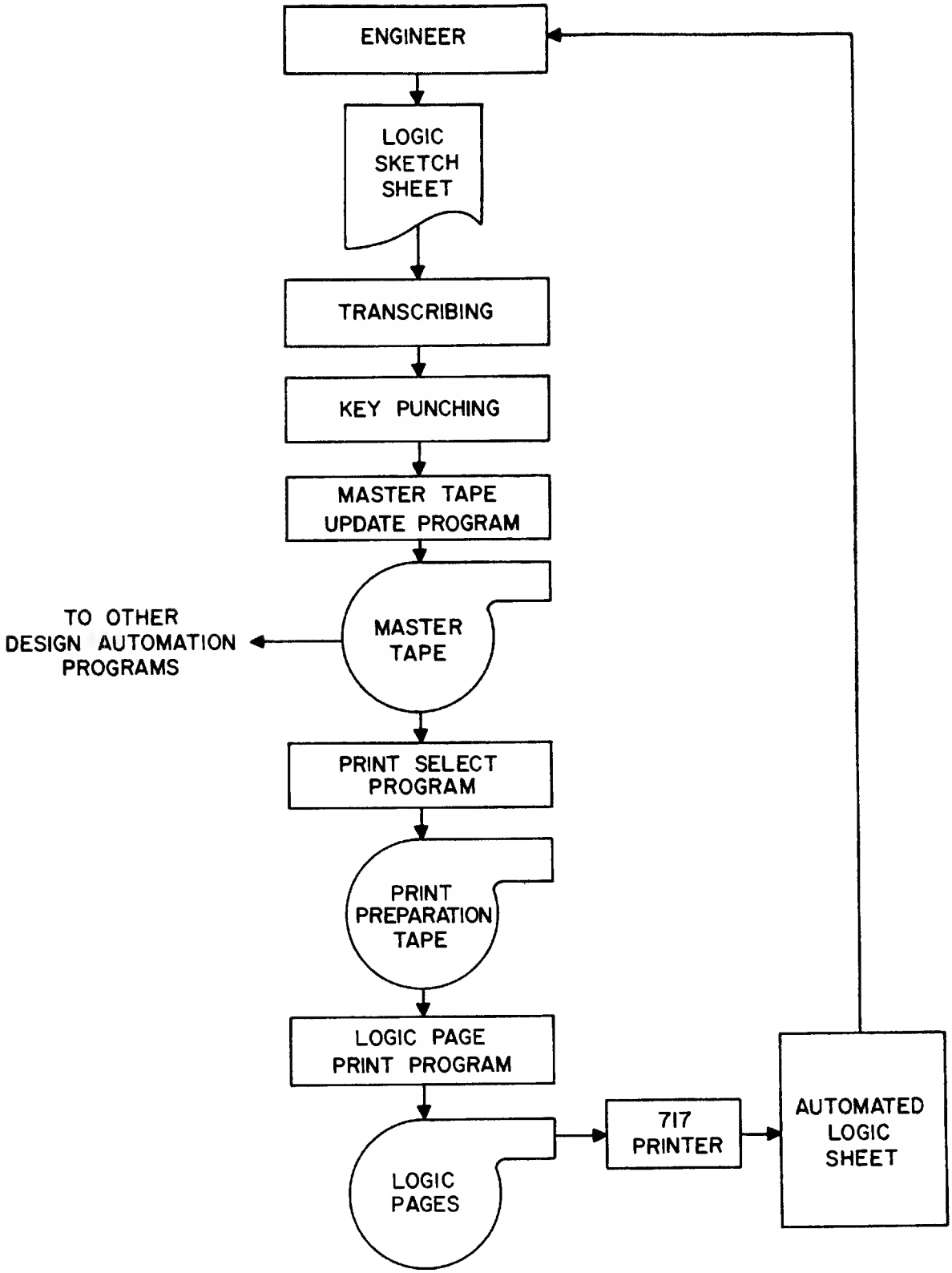


Figure 1. General Flow of Logic Diagram through the Design Automation System.

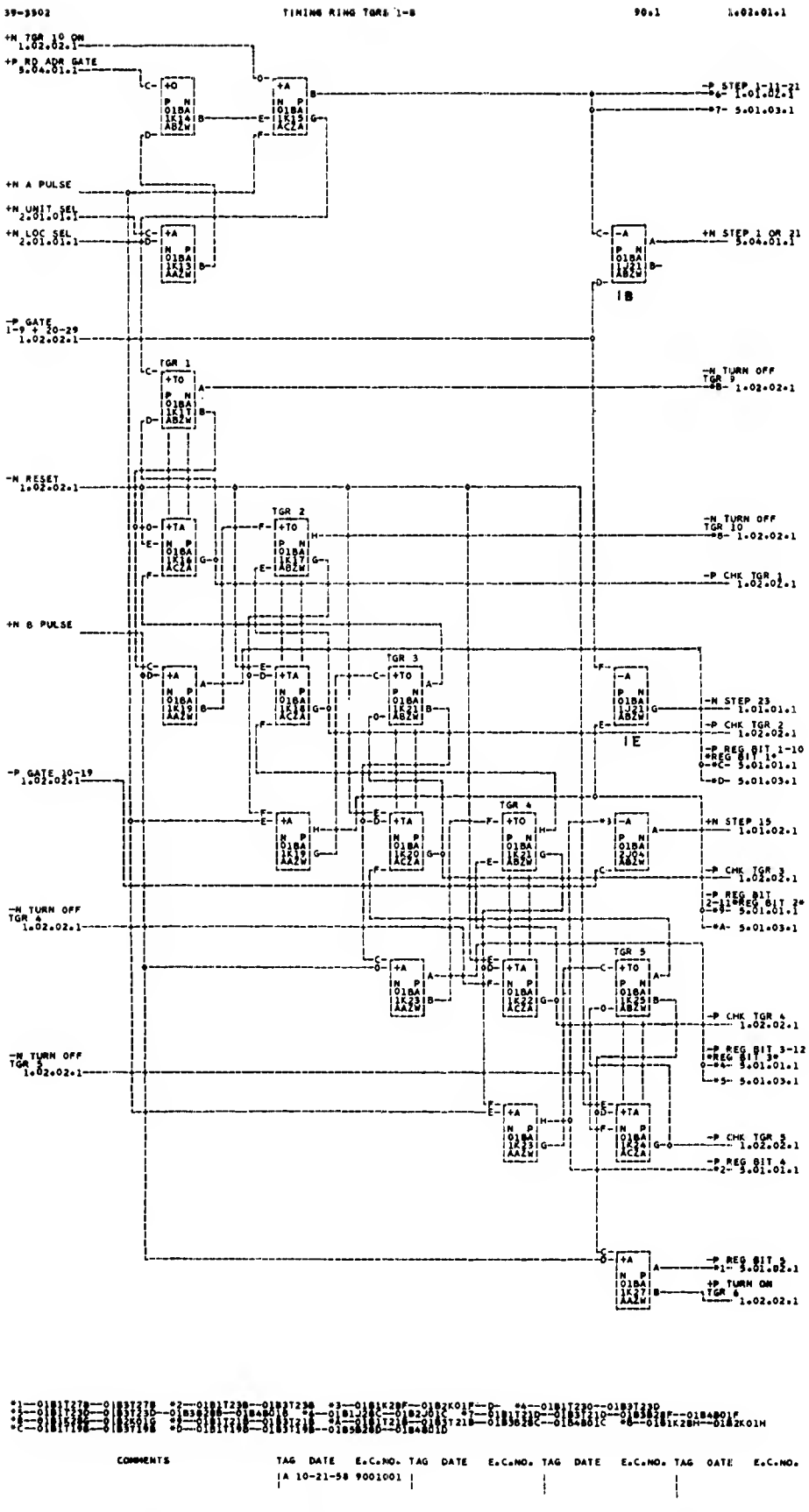
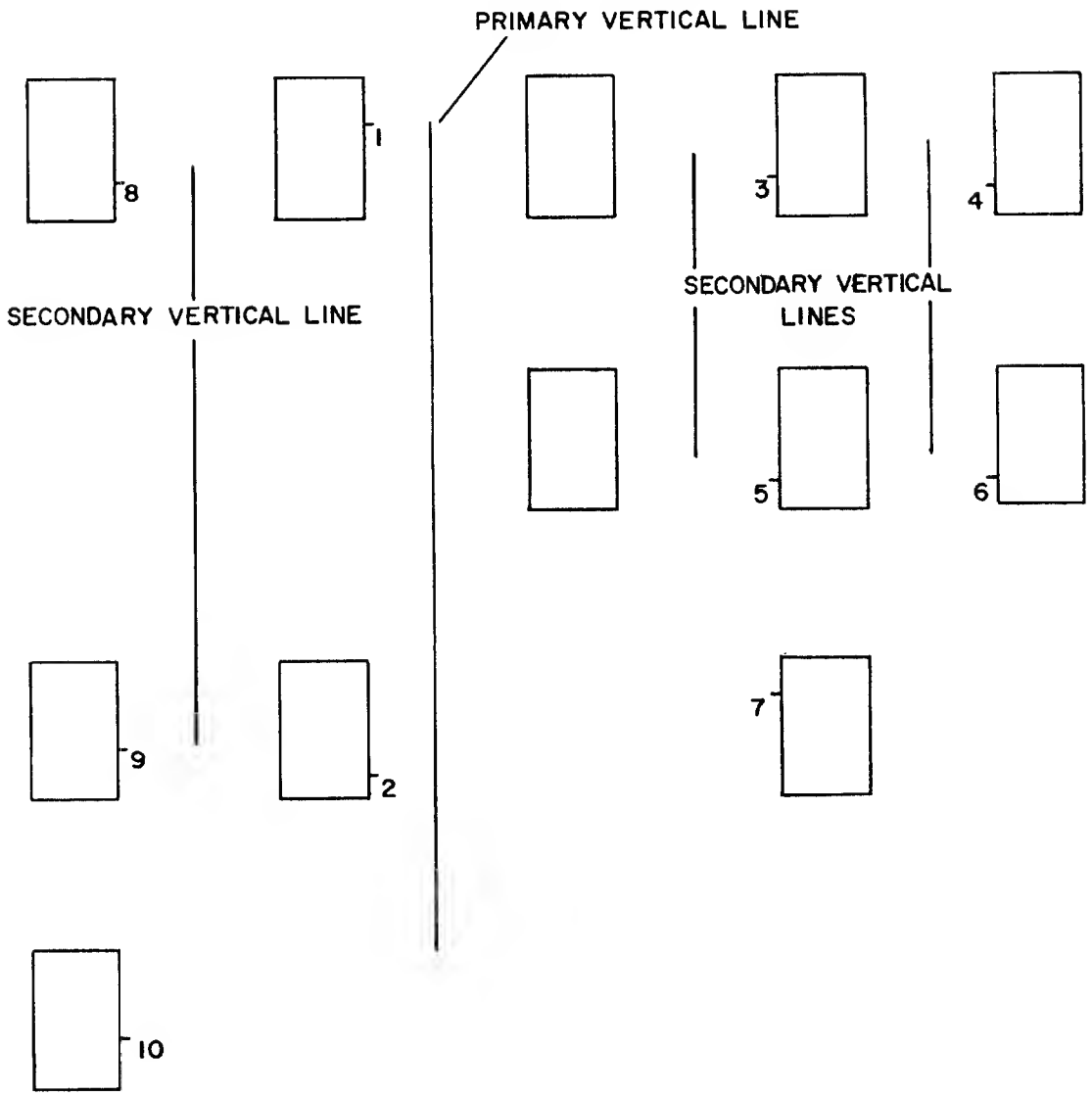


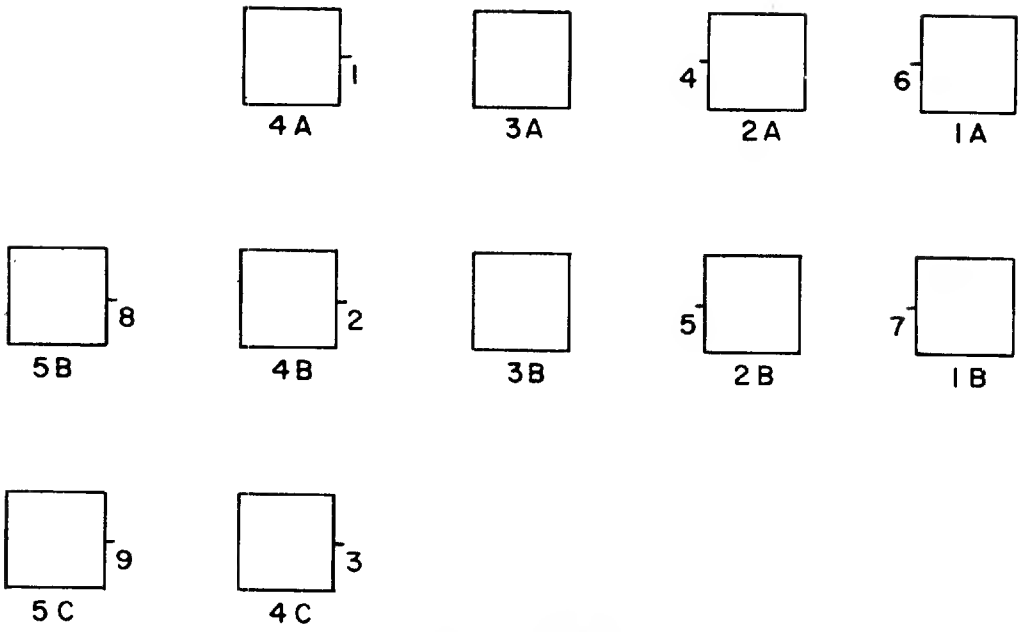
Figure 2. Machine-printed Systems Page.

EXAMPLE OF ROUTING DEFINITIONS FOR A NET

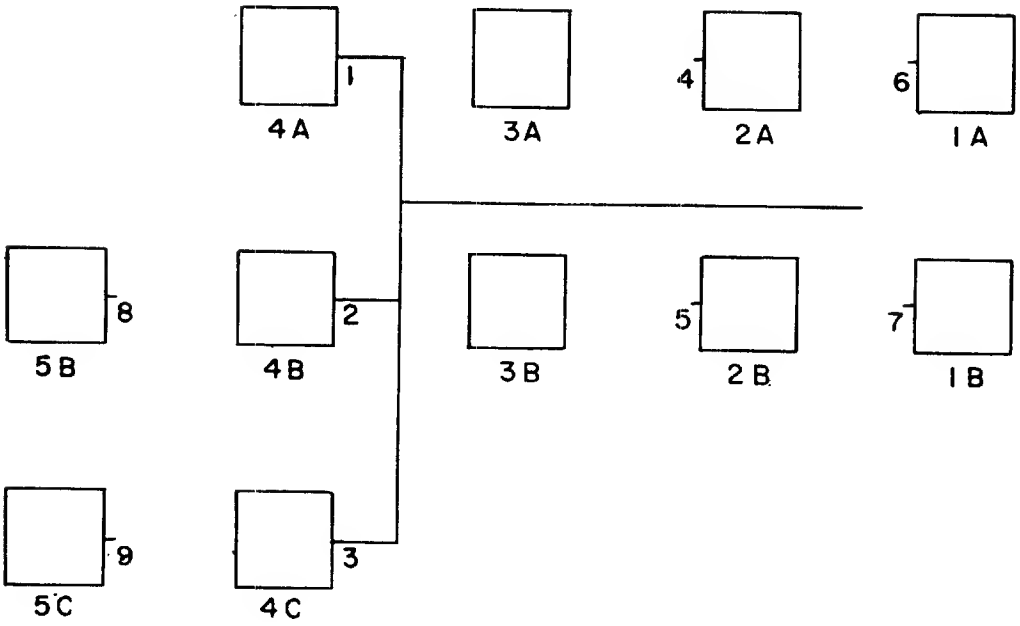


EXAMPLE # 1

SAMPLE OF LINE ROUTING

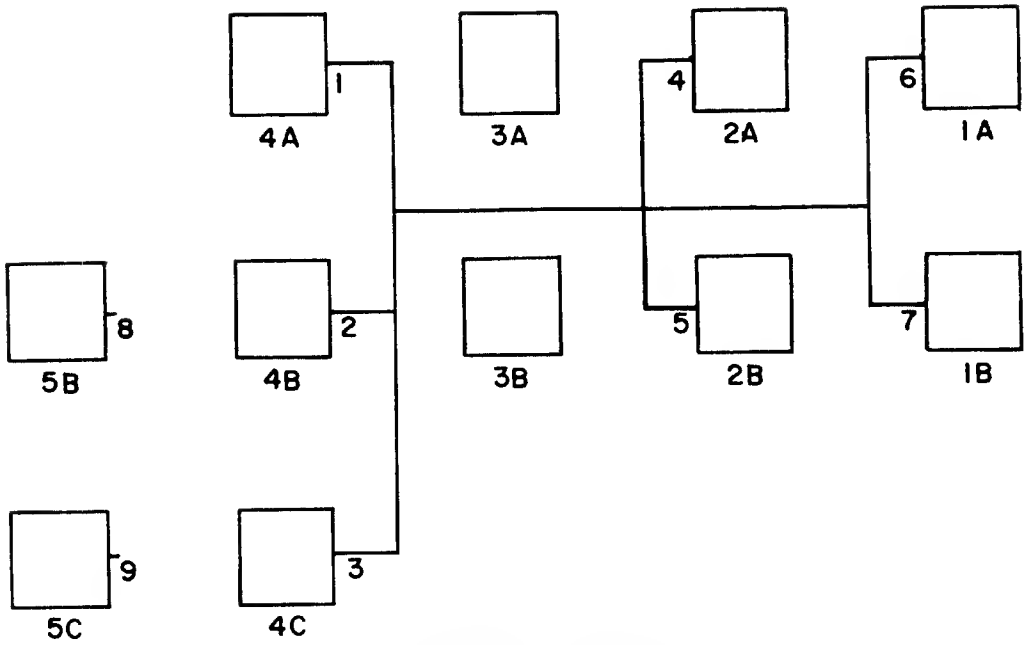


EXAMPLE # 2(a)

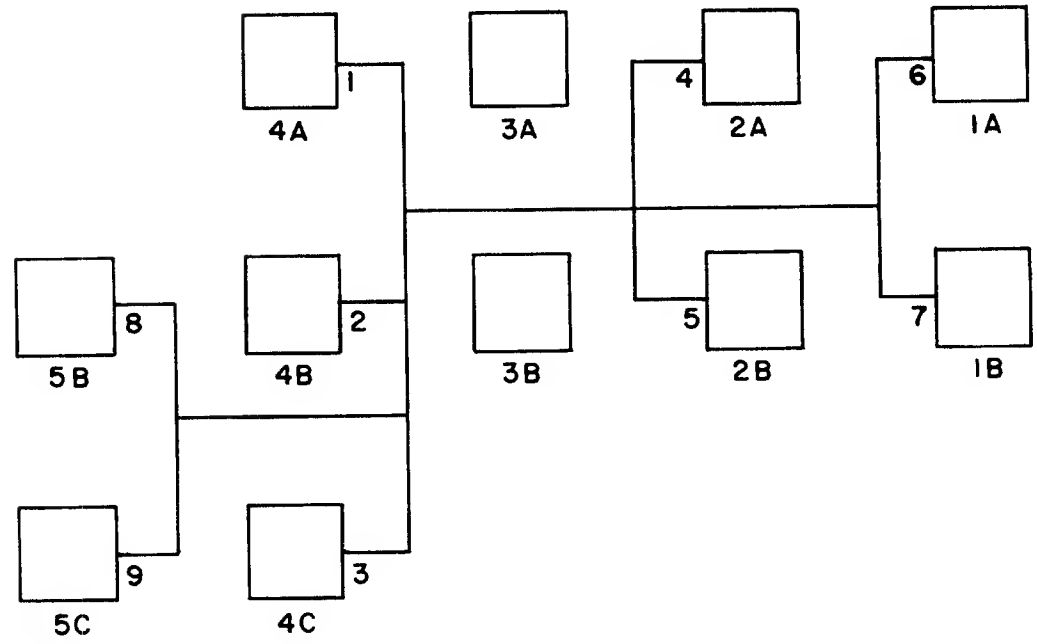


EXAMPLE # 2(b)

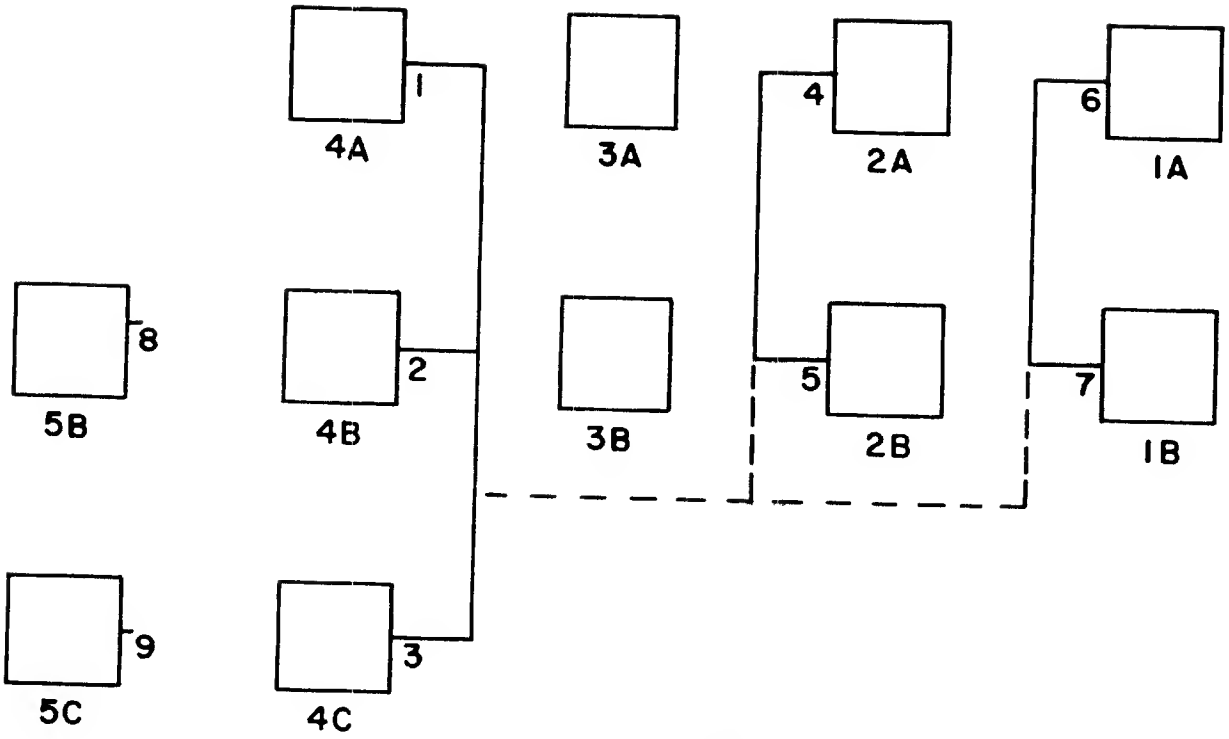
EXAMPLE OF LINE ROUTING (CONT.)



EXAMPLE # 2(c)

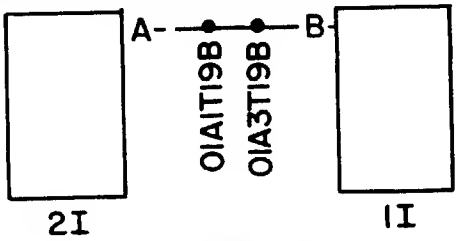


EXAMPLE # 2(d)



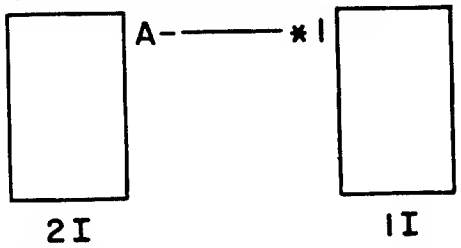
EXAMPLE # 2(e)

APPEARANCE OF EDGE CONNECTORS
ON LOGIC SKETCH SHEET



EXAMPLE # 3 (a)

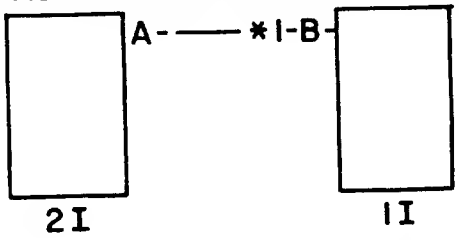
APPEARANCE OF EDGE CONNECTORS
IN PRINT PROGRAM MEMORY IMAGE



*1--01A1T19B--01A3T19B--B-

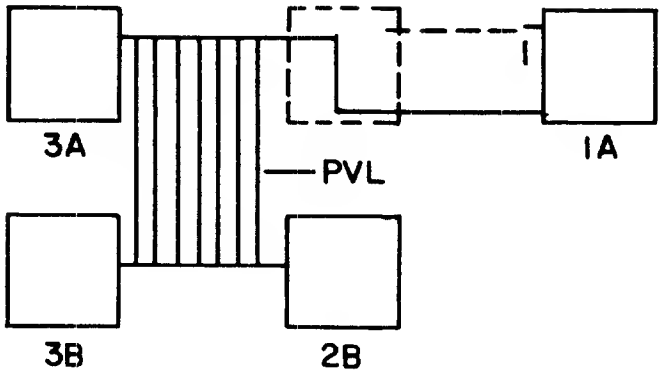
EXAMPLE # 3 (b)

APPEARANCE OF EDGE CONNECTORS
ON AUTOMATED LOGIC DIAGRAM

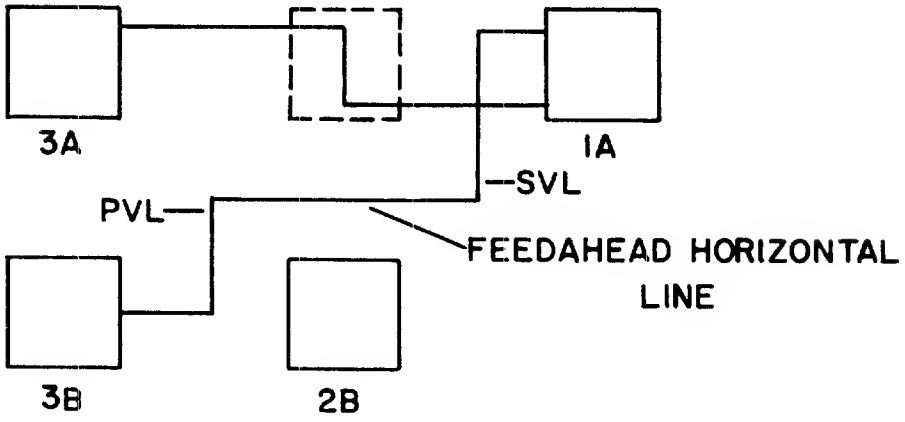


EXAMPLE # 3(c)

NET WHICH PROGRAM IS UNABLE TO ROUTE

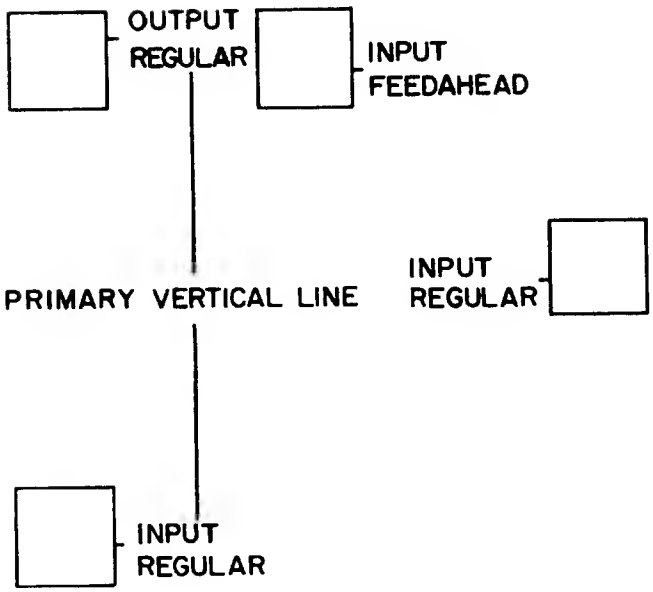


EXAMPLE # 4(a)

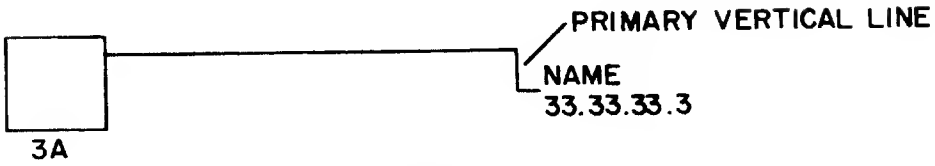


EXAMPLE # 4(b)

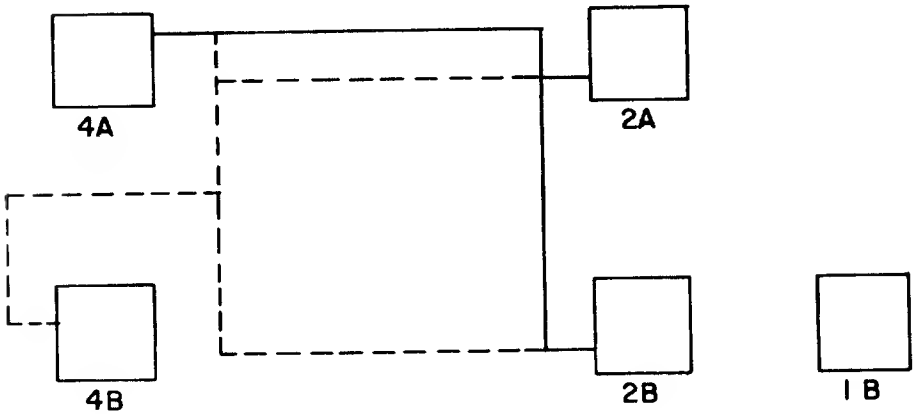
DETAIL DESCRIPTION OF ROUTING



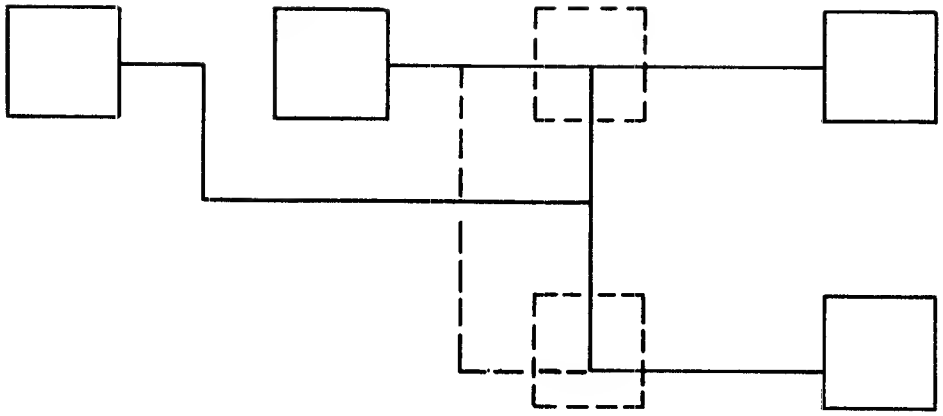
EXAMPLE # 5



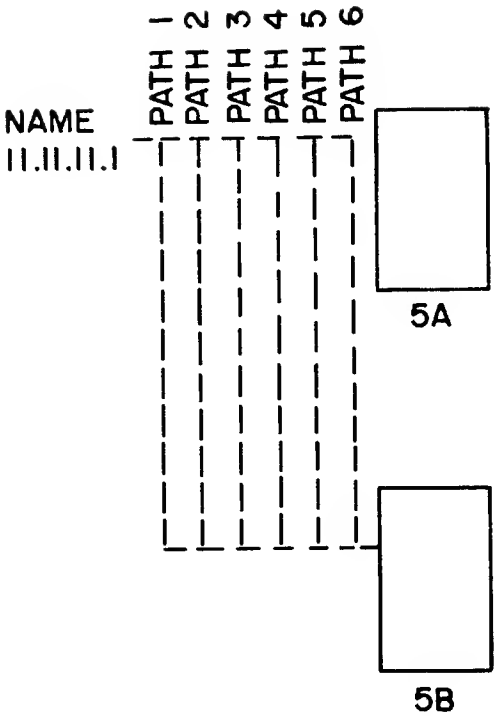
EXAMPLE # 6



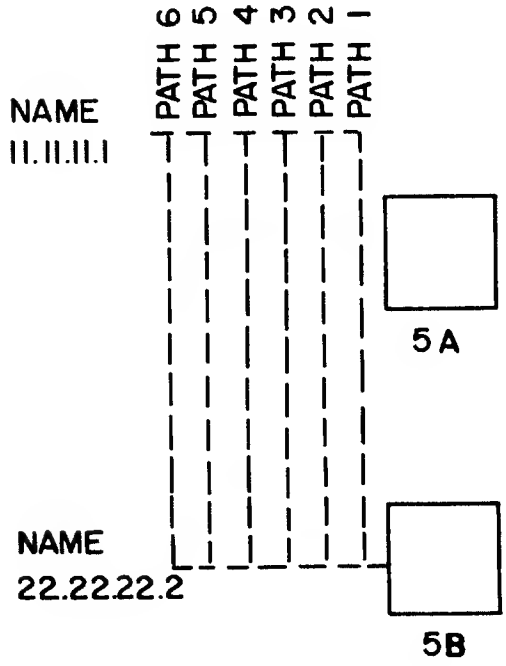
EXAMPLE # 7



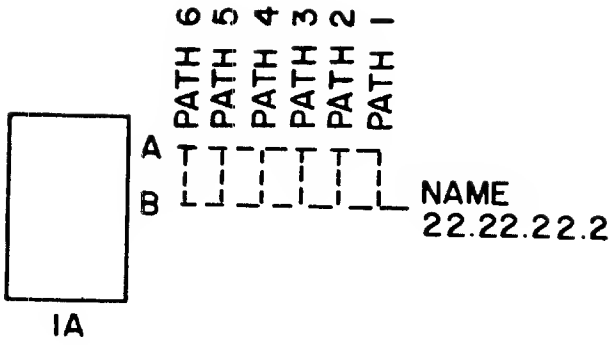
EXAMPLE #8



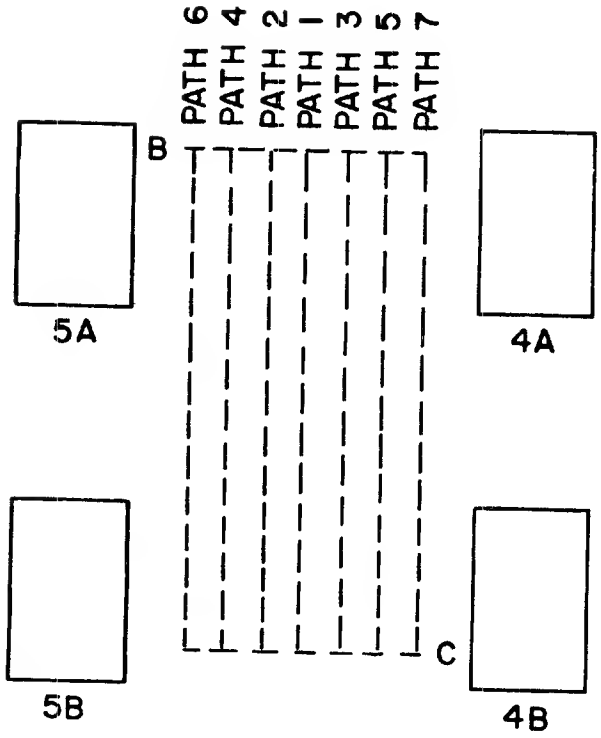
EXAMPLE # 9



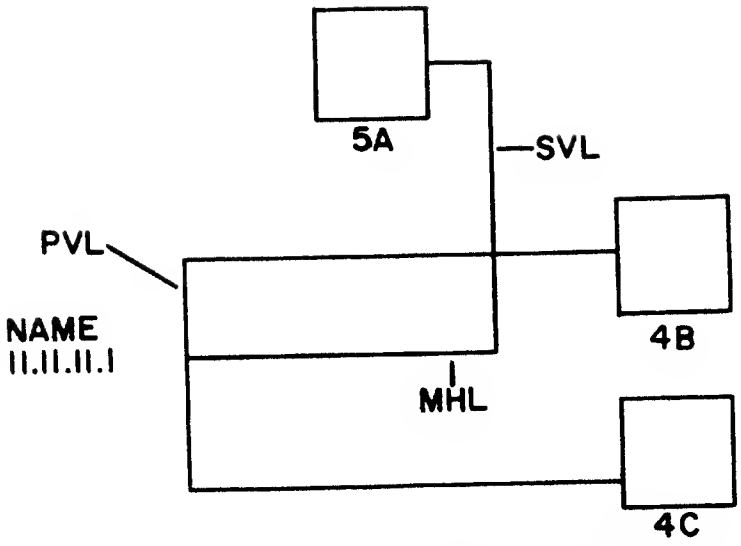
EXAMPLE # 10



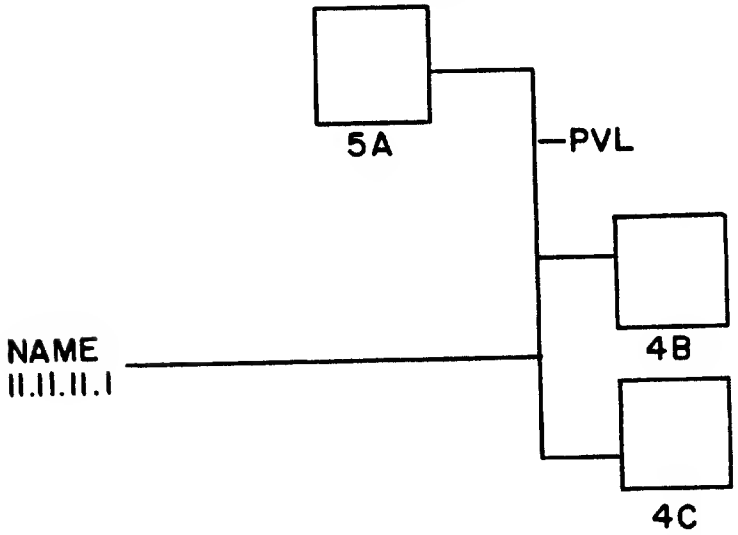
EXAMPLE # 11



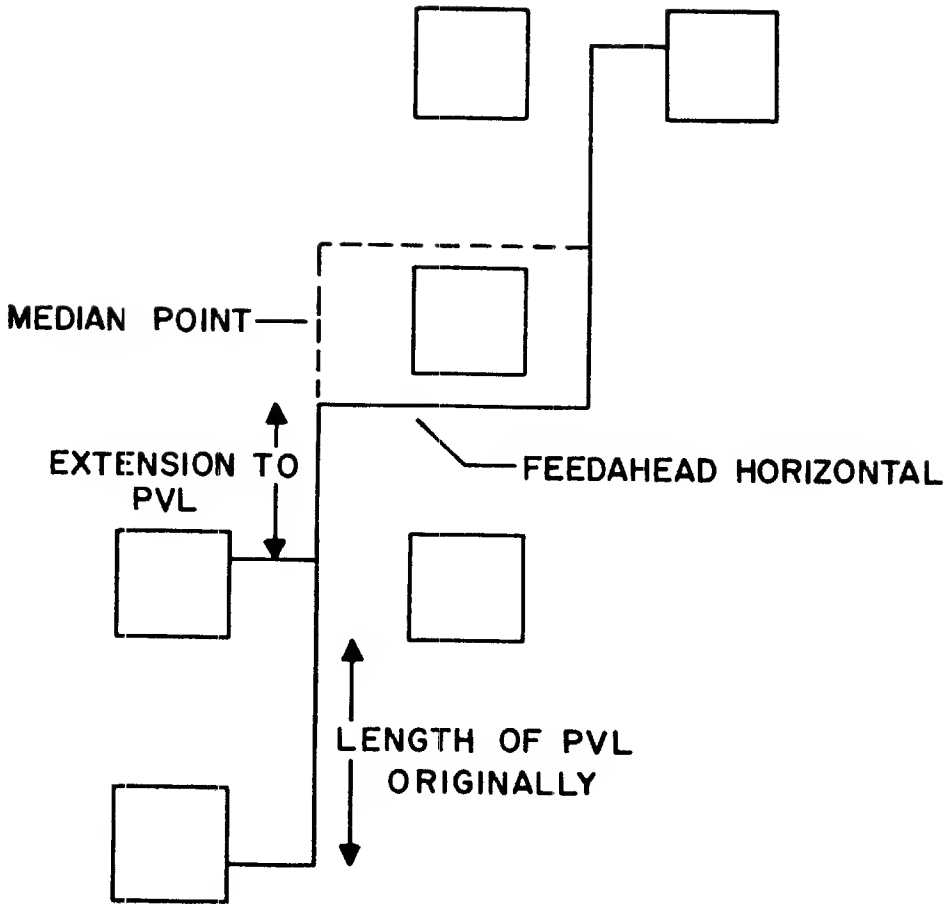
EXAMPLE # 12



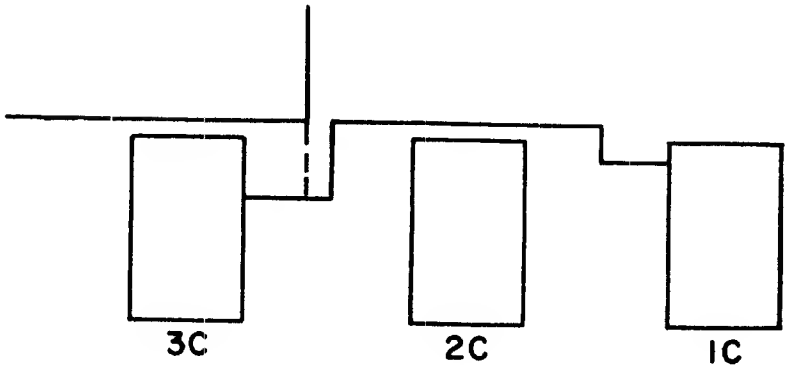
EXAMPLE # 13(a)



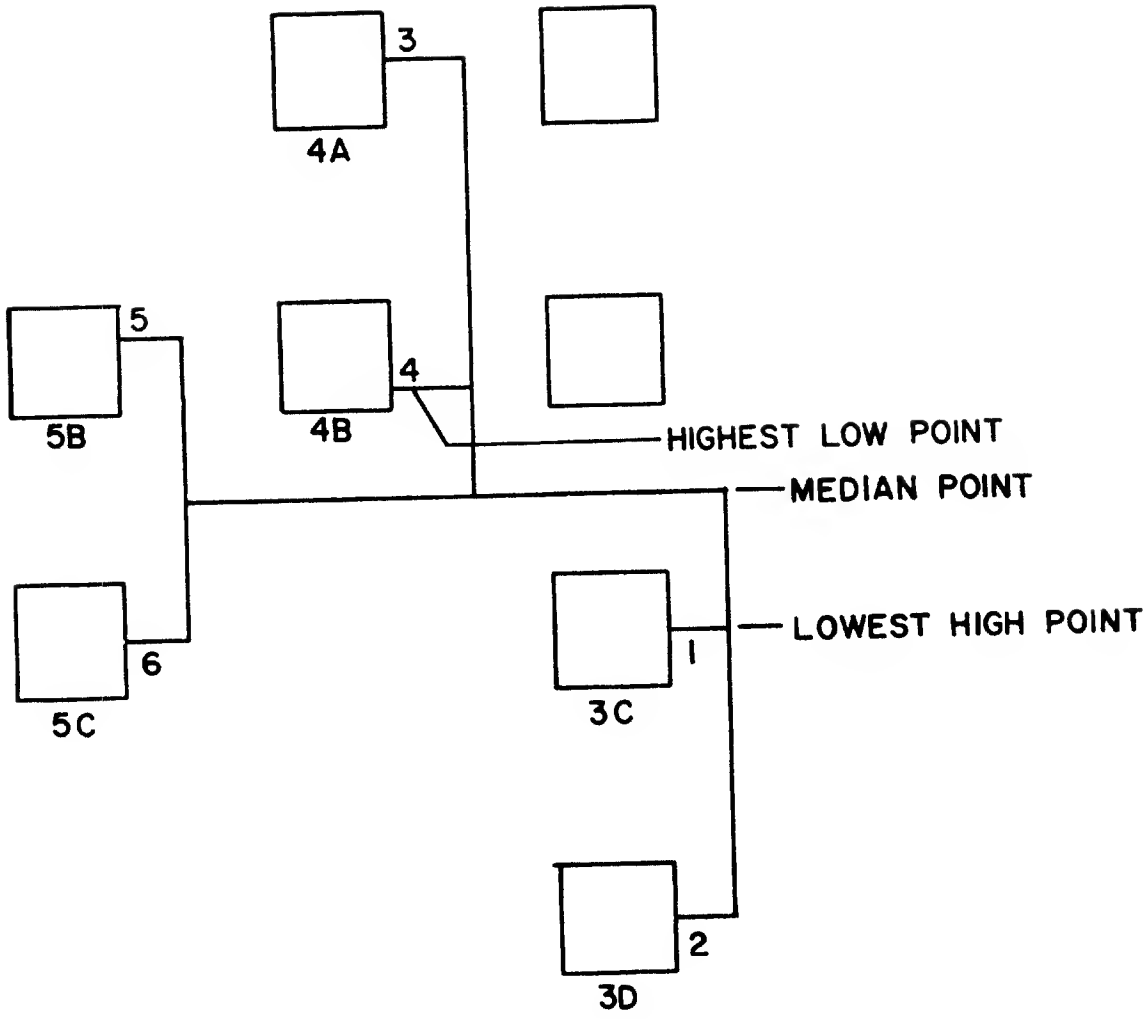
EXAMPLE # 13 (b)



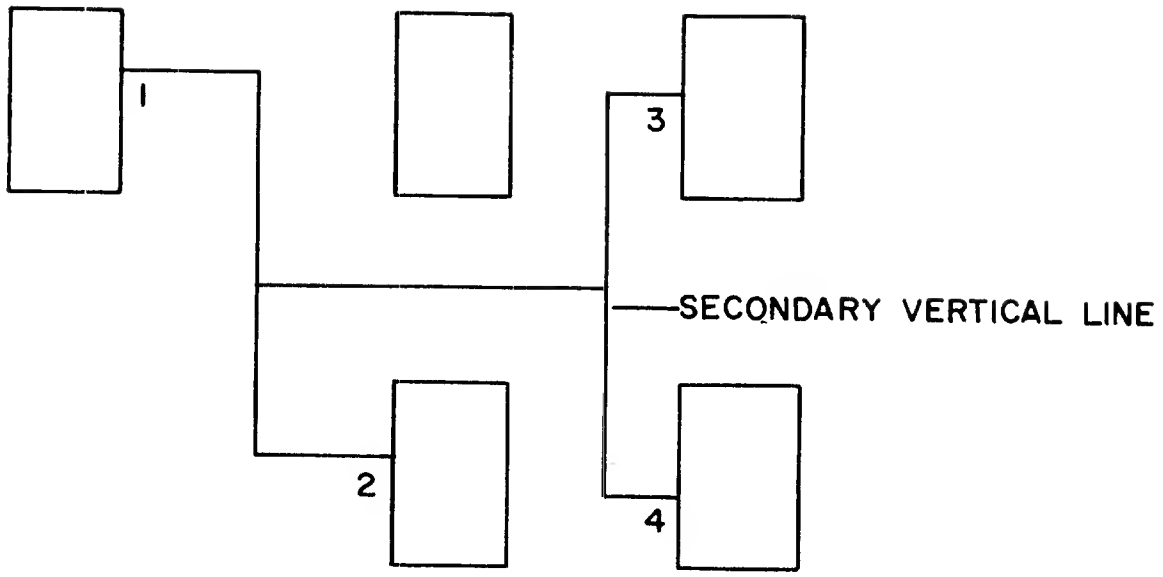
EXAMPLE # 14



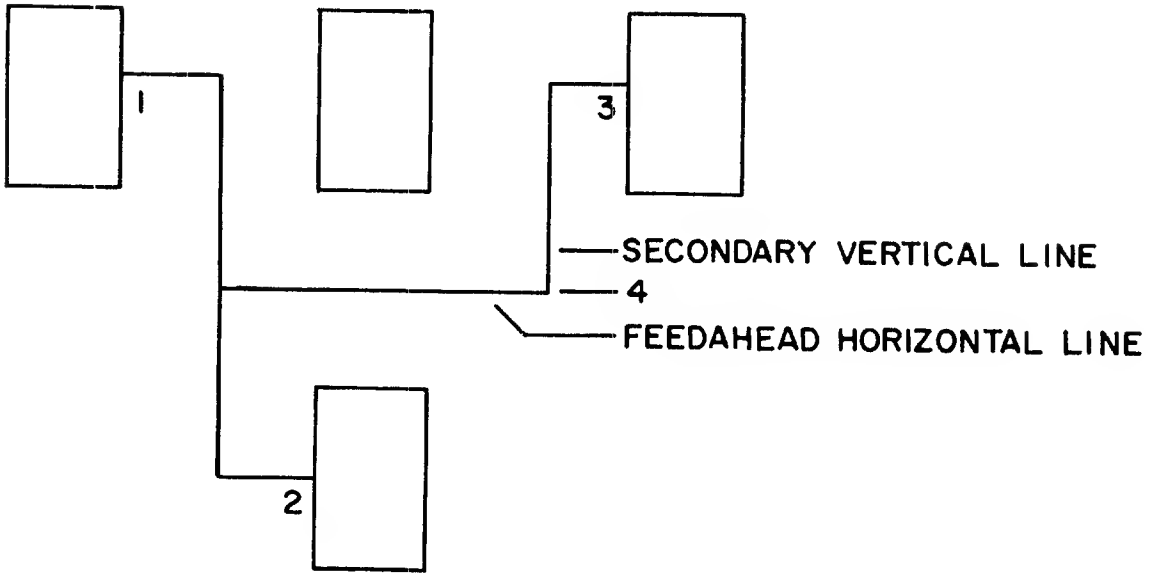
EXAMPLE #15


3
4A
5
5B
4
4B
HIGHEST LOW POINT
MEDIAN POINT
6
5C
1
3C
LOWEST HIGH POINT
2
3D

EXAMPLE # 16



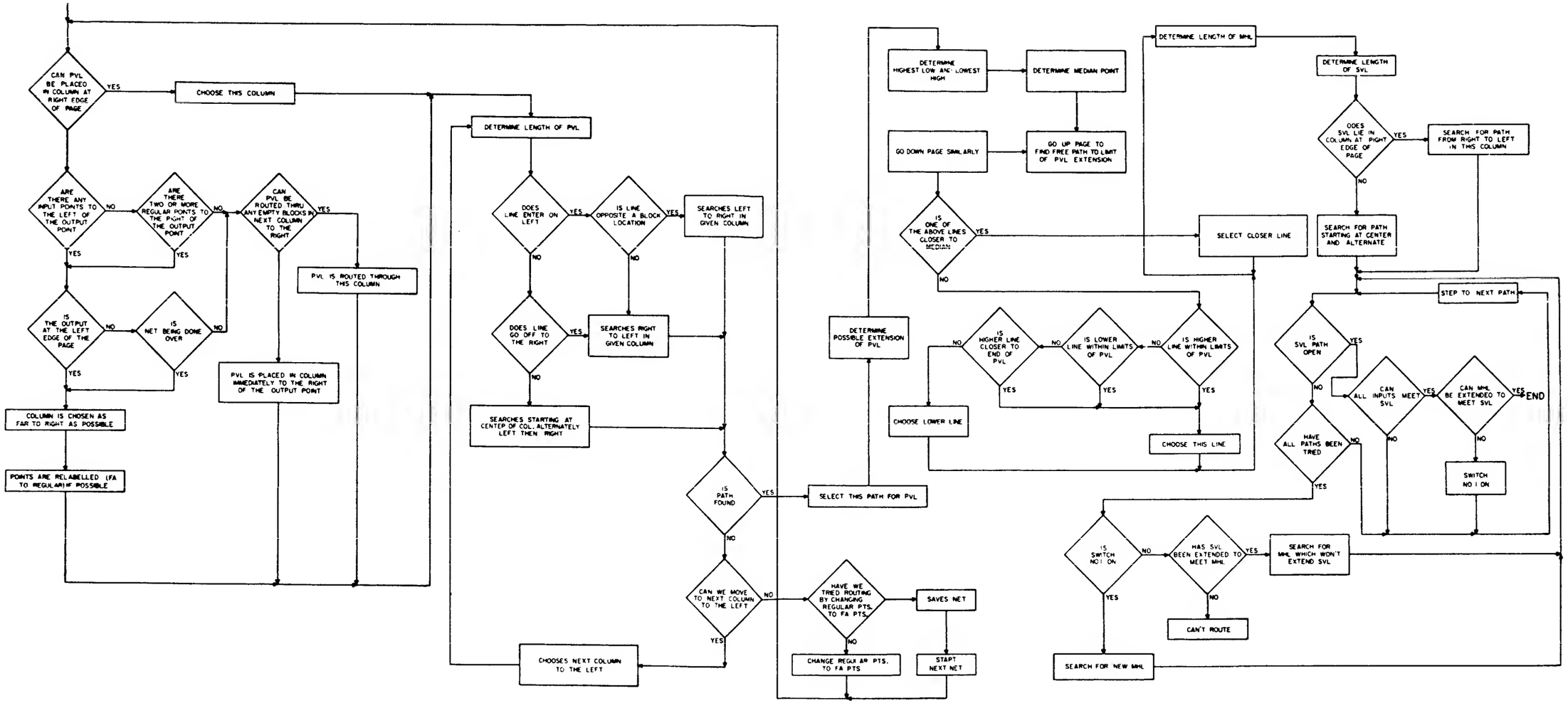
EXAMPLE # 17 (a)



EXAMPLE # 17 (b)

APPENDIX 2

Flow Chart of Line Routing Procedure


CAN PVL BE PLACED IN COLUMN AT RIGHT EDGE OF PAGE
YES
CHOOSE THIS COLUMN
ARE THERE ANY INPUT POINTS TO THE LEFT OF THE OUTPUT POINT
NO
ARE THERE TWO OR MORE REGULAR POINTS TO THE RIGHT OF THE OUTPUT POINT
NO
CAN PVL BE ROUTED THRU ANY EMPTY BLOCKS IN NEXT COLUMN TO THE RIGHT
YES
YES
YES
PVL IS ROUTED THROUGH THIS COLUMN
IS THE OUTPUT AT THE LEFT EDGE OF THE PAGE
NO
IS NET BEING DONE OVER
NO
YES
YES
PVL IS PLACED IN COLUMN IMMEDIATELY TO THE RIGHT OF THE OUTPUT POINT
COLUMN IS CHOSEN AS FAR TO RIGHT AS POSSIBLE
POINTS ARE RELABELLED (FA TO REGULAR) IF POSSIBLE
DETERMINE LENGTH OF PVL
DOES LINE ENTER ON LEFT
YES
IS LINE OPPOSITE A BLOCK LOCATION
YES
SEARCHES LEFT TO RIGHT IN GIVEN COLUMN
NO
NO
DOES LINE GO OFF TO THE RIGHT
YES
SEARCHES RIGHT TO LEFT IN GIVEN COLUMN
NO
SEARCHES STARTING AT CENTER OF COL. ALTERNATELY LEFT THEN RIGHT
IS PATH FOUND
YES
SELECT THIS PATH FOR PVL
NO
CAN WE MOVE TO NEXT COLUMN TO THE LEFT
NO
HAVE WE TRIED ROUTING BY CHANGING REGULAR PTS. TO FA PTS.
SAVES NET
YES
NO
CHOOSES NEXT COLUMN TO THE LEFT
CHANGE REGULAR PTS. TO FA PTS
START NEXT NET
DETERMINE POSSIBLE EXTENSION OF PVL
DETERMINE HIGHEST LOW AND LOWEST HIGH
DETERMINE MEDIAN POINT
GO DOWN PAGE SIMILARLY
GO UP PAGE TO FIND FREE PATH TO LIMIT OF PVL EXTENSION
IS ONE OF THE ABOVE LINES CLOSER TO MEDIAN
YES
SELECT CLOSER LINE
NO
IS HIGHER LINE CLOSER TO END OF PVL
NO
IS LOWER LINE WITHIN LIMITS OF PVL
NO
IS HIGHER LINE WITHIN LIMITS OF PVL
YES
YES
YES
CHOOSE LOWER LINE
CHOOSE THIS LINE
DETERMINE LENGTH OF MHL
DETERMINE LENGTH OF SVL
DOES SVL LIE IN COLUMN AT RIGHT EDGE OF PAGE
YES
SEARCH FOR PATH FROM RIGHT TO LEFT IN THIS COLUMN
NO
SEARCH FOR PATH STARTING AT CENTER AND ALTERNATE
STEP TO NEXT PATH
IS SVL PATH OPEN
YES
NO
ALL INPUTS MEET
YES
CAN MHL BE EXTENDED TO MEET SVL
YES
END
NO
NO
HAVE ALL PATHS BEEN TRIED
NO
SWITCH NO ! ON
NO
YES
IS SWITCH NO ! ON
NO
HAS SVL BEEN EXTENDED TO MEET MHL
YES
SEARCH FOR MHL WHICH WON'T EXTEND SVL
YES
NO
CAN'T ROUTE
SEARCH FOR NEW MHL

APPENDIX 3

Machine Configuration Required by the Logic Page Print Program

1. IBM 705 Model II Data Processing System

   a) C. P. U. - memory capacity of 40, 000 characters.
   b) 759 card reader control unit, 714 card reader.
   c) 754 tape control unit, four 727 III magnetic tape units.
   d) 757 printer control, 717 printer
   e) 734 magnetic drum,

2. IBM 705 Model III Data Processing System

   a) C. P. U. - memory capacity of 80, 000 characters.
   b) 759 card reader control unit, 714 card reader.
   c) 754 tape control unit, three 729 I magnetic tape units.
   d) 757 printer control, 717 printer.
   e) 734 magnetic drum.

APPENDIX 4

Print Program Running Time

1. For the IBM 705 Model II Data Processing Machine

$$T_t = 5 + \frac{3}{2} N \quad \text{sec/page}$$

Where N = number of nets on a page
K_{av} = average internal running time of the program
T_t = average running time per page.

2. For the IBM 705 Model III Data Processing Machine

$$T_t = 5 + \frac{2}{3} N \quad \text{sec/page}$$

APPENDIX 5

Input Tape Format (Pring Preparation Tape)

Record #	No of Characters	Description
1	80	tape label
2	4	number of characters in this record
	5	machine #
	8	logic page #
	7	part # of logic page
	2	block 01
	1	engineering change level (highest on page)
	33	page name
3	4	number of characters in this record
	2	block (1A, 8Z)
	1	EC tag or level
	1	block discontinued code
	4	circuit type
	4	machine features index
	4	voltage level
	4	physical location
	4	physical location
	4	part # of physical card
	1	number of inputs to the block
	2	configuration - a variable length description of the nets which originate from this block
n		there will be one record for each block on the page
n + 1	4	blank
	2	this record terminates the block information for this page (90^{+})
n + 2	732	engineers' comments and engineering change level information
n + 3		either an end of file or the first record of a new page
m		end of tape lebel

IBM

Prepared by:
Laboratory Publications, Development Laboratory, Data Systems Division
International Business Machines Corporation, Poughkeepsie, New York